Segmentation of Moving Objects Based on Connectivity Analysis of Weighted Region Adjacency Graph

A. Averbuch[1] O. Miller[2]
[1]School of Computer Science, Tel Aviv University
Tel Aviv 69978, Israel

[2]ArtiVision Technologies Pte Ltd
96 Robinson Road #13-02, Singapore

Abstract

The paper suggests a novel segmentation algorithm for separating moving objects from the background in video sequences without any prior information. The problem is formulated as a connectivity analysis of region adjacency graph (RAG) (see [1]) that is based on temporal information. By applying a watershed based algorithm, the video frame is segmented into a semantic homogeneous region. The boundary pixels in each region are compared with a series of consecutive frames in order to generate temporal information. The edges of the RAG represent temporal information. Each node represents a different homogeneous region. Analysis of the RAG's connectivity is achieved by modifying the breadth-first-search (BFS) algorithm. After a sufficient number of comparisons, each of the object's components is merged into a single segment which represents the moving object in the frame. The accuracy of the algorithm is proportional to the number of allowed comparisons.

Key words: Moving object, nodes connectivity, segmentation, spatio-temporal comparisons.

1 Introduction

Segmentation of moving objects (MO) aims at partitioning an image sequence into its physical moving objects and static background. The semantics of the moving object definition stems from the way humans analyze a video sequence. In general, it is agreed that humans analyze a video in terms of the objects of interest and their motions, where an object refers to a meaningful spatial and temporal region of a sequence. Despite the fact that the human visual system can easily distinguish between moving objects and background, robust video segmentation without any prior information is known

to be one of the most challenging problems in the field of video processing. Therefore, this problem has attracted the attention of many researchers and engineers.

Many applications related to image processing, video compression and pattern recognition rely on moving object segmentation and can utilize the new functionalities. For example, the ability to extract the moving objects of video surveillance systems may significantly reduce the number of false alarms caused by luminance changes. Algorithms for video indexing become a challenging problem to create fast and effective browsing, retrieval and management of visual databases. Current systems for video indexing and browsing are frame-based where each frame is analyzed by its global features such as its color histogram [2]. Therefore, events, which take place inside the frame, fail to be represented by a frame-based method. However, in object-based indexing, the object has to be identified. Low-level features such as color, shape, texture and motion, are available for the object's region and are attached to each object. Thus, an object-based indexing system can be much more detailed inside frame events and activities. In MPEG-4 [4] standard, a video sequence is considered to consist of independently moving objects and the encoding can be based on segmented objects. It also allows easy access to bitstreams of individual objects, manipulation of bitstreams and multiple use of content information by scene composition, which are all suitable for multimedia applications. Another important aspect is the immense popularity of the Internet and the WEB, which clearly demonstrate, that interactivity-based content is a key factor in many multimedia applications. Huge efforts have been invested over the last decade to find a solution to video object segmentation. In general, they can be broadly grouped into the following main categories.

Segmentation-based motion partitions the scene by its motion information. This information can be generated either by direct segmentation of a dense motion field, or by fitting a parametric motion model to regions. For example, a hierarchical structure segmentation approach is suggested in [6]. The idea of this approach is to iteratively refine the segmentation mask of a change detection connected region until maximum stability is achieved. A motion segmentation algorithm that breaks a scene into its most prominent moving groups was proposed in [8]. Instead of identifying the corresponding points between the frames, it suggests to find groups of pixels that are transformed from one frame to another. A segmentation algorithm which identifies uniform motion in the motion fields was suggested in [7].

Segmentation-based spatial morphological approaches were suggested by [9, 15, 18, 22]. An algorithm, which relies on parametric motion estimation of regions formed by an initial spatial segmentation, was suggested in [9]. The idea is to obtain a spatial partition by a pre-filtering using morphological open-close by a reconstruction operator. [15] suggested to use a structuring element and proposed a new design for marker extraction, which makes use of both luminance

and color information. A morphological segmentation algorithm, which utilizes morphological filters and watersheds segmentation as a basic tool, was presented in [18]. A statistical analysis of the watersheds algorithm to develop a multivalued morphological spatial segmentation method that incorporates an edge-driven marker extraction algorithm and a growing method, which integrates both color and edge information, was used in [22].

The goal of this paper is to present a robust algorithm to segment moving objects. We introduce a temporal segmentation, which compares several successive frames with a single reference frame. Each comparison provides a new temporal information about the motion relative to the reference frame. We use a spatial segmentation, which is based on the watershed algorithm and a merging process, as an initial segmentation. Then, an iterative algorithm (which is called Nodes Connectivity Analysis) analyzes the spatial and the temporal information on a regions adjacency graph that is obtained from the spatial segmentation. Each iteration, extracts a set of nodes, which are candidates to represent the moving objects regions. After sufficient iterations these sets become similar in theirs shape and no additional iteration is needed. The performance of the algorithm is proportional to the number of allowed iterations.

The paper is organized as follows. Section 2 provides a short description of the proposed algorithm followed by an outline of the main processes. Section 3 presents the pre-processing step of the algorithm, which is the initial still segmentation algorithm. Section 4 describes the temporal information generator based on several frames comparisons. A presentation of the tempo-spatial connectivity analysis is given in section 5. Section 6 describes the connectivity analysis along with the temporal generator to extract the moving object. Experimental results are given in section 7.

2 Outline of the algorithm

A common framework to achieve moving object segmentation consists of three main phases: temporal segmentation, spatial segmentation and a hybrid of both. Temporal segmentation aims at detecting the moving parts in the sequence, spatial segmentation divides the frame into semantic homogeneous regions. Combination of spatial and temporal segmentation [7, 9, 11] produces a moving object segmentation mask.

The proposed method represents the spatial segmentation information that is given by nodes of a weighted Region Adjacency Graph (RAG), which was introduced in [1], such that each homogeneous region is assigned to a different node. The edges of the RAG are constructed by temporal information. The temporal analysis is based on multiple iterations of a change detection technique such that each iteration represents the intensity changes between a reference frame I_t and a successive frame in the sequence $I_{t+i}, i = 1, \ldots, N_t$. The size of N_t depends on the reference frame. Thus, each iteration in

the temporal phase provides updated weights for the edges in the RAG. The spatial and temporal information are combined by the Nodes Connectivity Analysis (NCA) algorithm, which is applied after each iteration in the temporal phase to an updated RAG. Each NCA application extracts a set of candidate nodes that represents the moving object regions that correspond to the current iteration. During the applications of the NCA, the object's set of nodes is converged into similar groups of nodes. The algorithm is terminated and extracts the object set of nodes from the last NCA iteration. Figure 2.1 depicts the main components of the entire segmentation algorithm, where I_t is the reference frame from which the object was extracted and I_{t+i} $(i > 0)$ is a successive frame that belongs to the current scene.

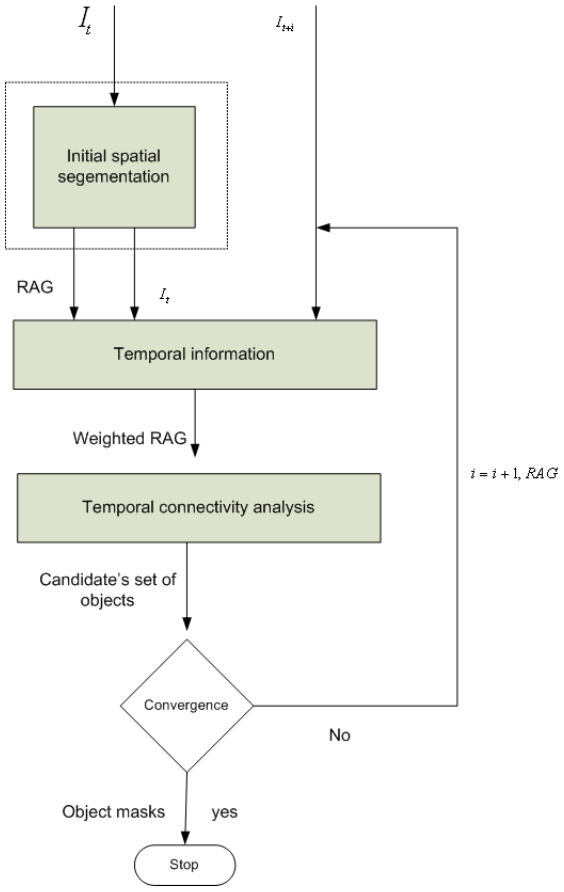


Figure 2.1: The main steps in the moving object segmentation algorithm. Increase of i by 1 brings a successive frame I_{t+i} into the temporal step

3 Initial Spatial Segmentation

Spatial segmentation, which is considered here as a preprocessing step, partitions the image into semantic homogeneous regions. The regions are distinguished by their encompassing boundaries that

were obtained from spatial segmentation. The watershed algorithm ([20]) is a very popular method to perform a fast spatial segmentation. However, its output usually results in over segmentation.

Our motivation in having an initial spatial segmentation is to achieve a minimal number of segmented regions while preserving the homogeneity criteria of each region. For this purpose, we apply the segmentation algorithm that is described in [19] as the initial spatial segmentation step. This algorithm combines edge and region-based techniques through the morphological algorithm of watersheding. It uses the output of the watershed transform as the starting point for a bottom-up hierarchical merging approach, where at each step the most similar pair from adjacent regions is detected and merged. Figure 3.1 depicts the main steps in the initial spatial segmentation for the moving object segmentation.

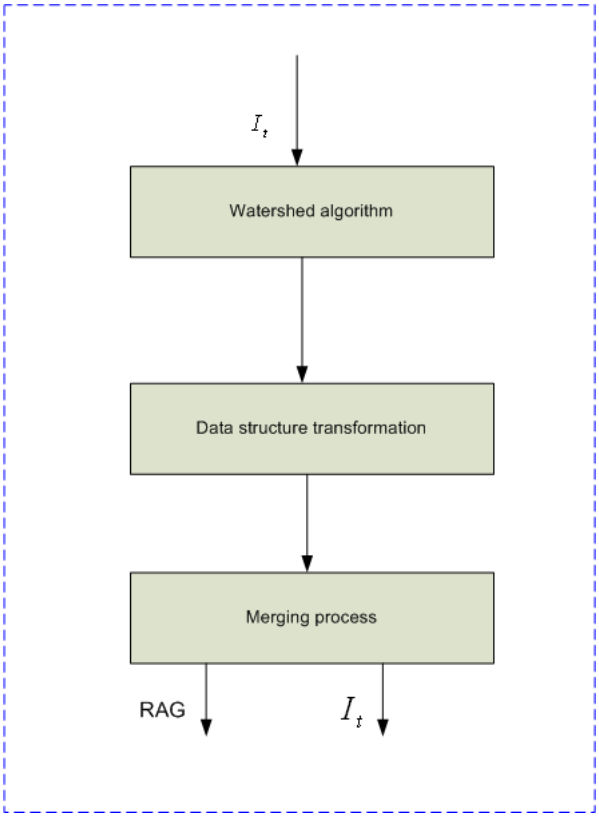


Figure 3.1: Flow of the initial spatial segmentation phase. This is considered as a preprocessing step for the entire algorithm. The output of this algorithm is an *unweighted undirected* region adjacency graph such that each node represents a different semantic homogeneous region in the reference frame I_t to be segmented.

After applying the still image segmentation algorithm [19], a transformation of its output into a RAG data structure is required for the rest of the spatio-temporal segmentation process. For the n segmented regions, we define the undirected graph $G = (V, E)$, where $|V| = n$ and $e(i, j) \in E$. The

nodes of G represent the segmented regions and an edge $e(i,j) \in E$ represents two adjacent regions i and j. The edge $e(i,j)$ contains all the adjacent boundary pixels of i and j. Hence, all the boundaries' segmentation are represented by E. Figure 3.2 shows the transformation of a still image segmentation (Fig. 3.2a) into an RAG $G = (V, E)$ (Fig. 3.2b) - see [1]. The edges $e(1,2), e(1,3), e(1,4)$ and $e(1,5)$ in Fig. 3.2b denote the four neighbors of the region R_1 in Fig. 3.2a.

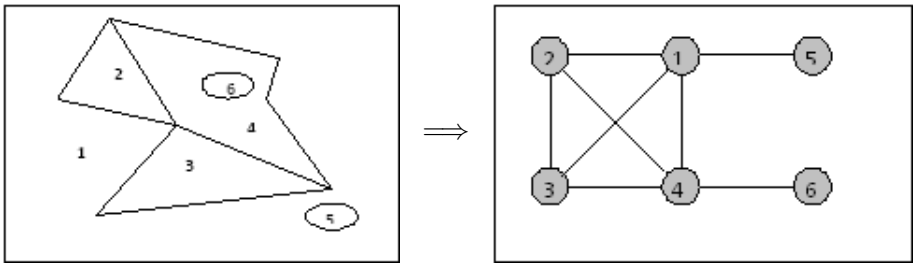


Figure 3.2: Illustration how the region-based segmentation is transformed into a RAG data introduced in [1]

4 Temporal Information

Temporal information is needed to achieve moving object segmentation. It can be obtained by either motion detection or change detection techniques. Motion detection identifies which pixels or regions have moved between two instants. In general, motion causes intensity changes in the pixel magnitudes. Therefore, it is the main cue for locating the moving objects. Intensity change is necessary for motion detection but not sufficient. On the other hand, intensity changes may be misleading since they can come from other sources besides motion such as camera noise and illumination changes. Moreover, intensity differences produced by the camera and illumination changes may be stronger than those produced by motion. Practically, this may occur when the moving object is characterized by a weak texture. In these cases, the motion information generates errors in motion estimation. Since we are interested only in the changes of an object rather than its motion estimation, we choose to obtain the temporal information by focusing on the intensity change attributes between frames. This is considerably faster than computing motion estimation attributes.

One of the main problems in achieving reliable temporal information regarding an object's location is the fact that the object or part of it can be static in more than two consecutive frames. Therefore, to overcome this problem, we will introduce in section 4.3 a change detection based approach that compares several frames with a single reference frame in which we wish to extract its MO. A probability for change is assigned to each edge in the RAG (section 4.4). It is calculated on accumulated

information by a statistical methodology after the completion of each comparison. If the object is static during the comparison process, it cannot be detected. This approach guarantees that after a sufficient number of iterations the object will no longer remain in a static position.

Change detection techniques assume that the input frames are registered. Such an assumption is generally not realistic, since most of the video sequences are captured using a moving camera, and thus, consecutive frames are not registered. Therefore, to use a moving camera, the motion induced by the camera must be detected and compensated by a pre-processing step. For this purpose, global motion estimation and compensation (section 4.1 - see also [2, 3]) are used before the application of the change comparison phase and before a scene cut detection (section 4.2) is called to ensure that the frames belong to the same scene. Figure 4.1 depicts the entire temporal phase.

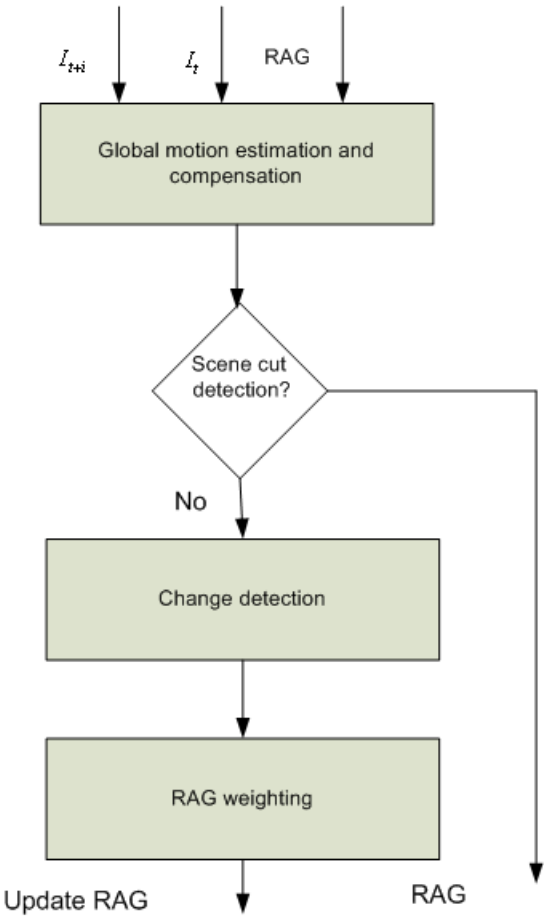


Figure 4.1: Processing of the temporal information is processed. I_t is the reference frame, and $I_{t+i}, i = 1, \ldots, N$, is the current i^{th} frame that is compared to I_t

4.1 Global Motion Estimation and Compensation

The goal is to compensate the camera's movements by a pre-processing step in order to convert the background to a static area. This is done by finding the motion between two frames. We use a

differential parametric optical flow estimation [23]. The camera motion is represented by a 8-parameter perspective motion model that is defined by:

$$\begin{pmatrix} x'_j \\ y'_j \end{pmatrix} = \begin{pmatrix} \frac{a_1+a_2x_j+a_3y_j}{a_7x_j+a_8y_j+1} \\ \frac{a_4+a_5x_j+a_6y_j}{a_7x_j+a_8y_j+1} \end{pmatrix}. \tag{4.1}$$

We want to minimize the squared sum intensity errors between the two given images I_t and I_{t+i}:

$$ssd = \sum e_j^2 \quad e_j = I_{t+i}(x'_j, y'_j) - I_t(x_j, y_j), \quad i = 1, \ldots, N_t \tag{4.2}$$

where (x_j, y_j) denotes the spatial coordinates of the j^{th} pixel in the current frame, (x'_j, y'_j) denotes the coordinates of the corresponding pixel in the reference frame. Since we are interested only in the motion of the background, which is considered to be planar, the perspective motion model fits this assumption.

The global motion estimation technique ([2, 3, 23])is applied within a hierarchical framework. Three multiscale pyramids are constructed by subsampling the current frame twice, using a 3-tap low-pass filter with the coefficients $[0.25, 0.5, 0.25]$. The global motion estimation proceeds from the coarsest level of the pyramid to the finest level. Several iterations are performed at each level. The minimization is performed using the Levenberg-Marquardt non-linear minimization algorithm. Since this method may not converge in the presence of large displacement, therefore, to assure convergence the starting point of the minimization should be within the basin of the global minimum. To achieve this, a coarse estimate of the translated component of the displacement is computed. The initial conditions are obtained by a matching technique applied at the top level of the pyramid using a logarithmic three-step search.

The Levenberg-Marquardt algorithm requires computation of the partial derivatives of e_i (Eq. 4.2) with respect to the unknown motion parameters $\{a_1, \ldots, a_8\}$. Using these partial derivatives, the algorithm computes an approximated Hessian matrix (H) and weighted gradient vector (B) with the components

$$\begin{aligned} h_{kl} &= \textstyle\sum_j \frac{\partial e_j}{\partial e_k} \frac{\partial e_j}{\partial a_l} \quad k, l = 1, \ldots, 8 \\ b_k &= -\textstyle\sum_j e_j \frac{\partial e_j}{\partial a_k} \quad k = 1, \ldots, 8 \end{aligned} \tag{4.3}$$

and then updates the motion parameter estimate by $\Delta a = H^{-1}b$. The Levenberg-Marquardt algorithm is first applied at the top level of the pyramid and it is iterated until a suitable convergence test is met. The resulting motion parameters are projected onto an intermediate level of the pyramid and the Levenberg-Marquardt algorithm is iterated again. Finally, the motion parameters are projected onto the base level of the pyramid and the Levenberg-Marquardt algorithm is iterated to produce the final motion parameter.

However, the global motion estimation may falsely detect small global motions due to local motions. In order to prevent this, the sum of the estimated motion parameters $\{a_1, \ldots, a_8\}$ is compared to a

threshold T, and it is compensated only when $(\sum_{i=1}^{8} a_j) - 2 > T$ by warping frame I_t to frame I_{t+i} according to the perspective projection in Eq. (4.1).

4.2 Scene-Cut Detection

The scene-cut detection method cuts the video into a number of sequences where the operation in section 4.1 can take place. If a new scene is detected then no more iterations in the temporal phase are needed. In general, a new scene is defined if the average absolute difference (MAD) of the gray levels between two consecutive frames I_t and I_{t+1} is greater than a certain threshold. However, for scenes with a lot of camera movements or scenes with several moving objects, a measure that takes into account the MAD between frames inside the scene and frames between the scenes is given by $SMAD_t = MAD_t - MAD_{t-1}$ where MAD_t is the MAD between I_t and I_{t+1} and MAD_{t-1} is the MAD between I_t and I_{t-1}. The $SDMAD$ is the second derivative of a frame and exhibits large positive and negative peaks in frames with scene changes. Due to large positive and negative peaks in the preceding and succeeding scene-changed frames, the detection performance of a scene change is insensitive to the selected threshold value. The $SDMAD$ may fluctuate in small positive and negative values around zero when a scene change does not occur.

4.3 Multiple Comparisons (MC) Using Change Detection Techniques

Change detection technique to obtain temporal information is a common approach. However, change detection techniques suffer from two critical drawbacks. First, unless the object is sufficiently textured, the interior of the object will remain unchanged even if the object has moved. Second, a change detection algorithm extracts regions of change relatively to the compared frame, which includes covered background, while object extraction does not include these regions. In addition, temporal information, which is based on two frames without considering historical information, may fail to detect objects whose local positions are temporally static. To overcome the above limitations and to gather essential information about the object in the sequence, an accumulated analysis of more than two consecutive frames is needed.

Assume I_t and I_{t+i} are frames from the same scene where $i = 1, \ldots, N_t$ represent the lags between the frames. We propose a multiple comparisons (MC) approach, based on a change detection algorithm suggested in [21], to compare between the pairs $\{(I_t, I_{t+1}), (I_t, I_{t+2}), \ldots, (I_t, I_{t+N_t})\}$ as the main cue for the objects' movements in I_t. In other words, the frames I_{t+i}, $i = 1, \ldots, N_t$, are compared to the reference frame I_t. The comparison result between each pair is a change detection mask (CDM). Each CDM is treated as an accumulated information from previous comparisons. The following formalizes the MC approach and show how to construct the CDMs.

Denote an object's set of pixels by $O_t \in I_t$ where $O_t = \{(x, y) | x, y \in obj_t\}$ and obj_t represents

the segments of the moving object in I_t. After i consecutive frames we have $O_{t+i} \triangleq \{x + s_x^{t,t+i}, y + s_y^{t,t+i} | x, y \in obj_t\}$ where $s_x^{t,t+i}$ and $s_y^{t,t+i}$ are the shifts between I_t and I_{t+i} produced by the motion vectors of obj_t. After i comparisons between I_t and I_{t+i}, both O_{t+i} and O_t will be considered as 'change' in the $CDM_{t,t+i}$ output where $CDM_{t,t+i}$ represents the comparison results between I_t and I_{t+i}. In addition, each $CDM_{t,t+i}$, $i = 1, \ldots, N_t$, will consider the set O_t as a 'change' in its fixed location, while the set O_{t+i} is considered as 'change' in its dynamic location corresponding to the object's location in I_{t+i}. Practically, such a comparison caused the set O_t to be represented as an uncovered area for every $i = 1, \ldots, N_t$ and the set O_{t+i} to be represented as an occlusion area. The fact that O_t remains in its fixed location during i comparisons together with spatial segmentation will be utilized to find the moving object in I_t. Figure 4.2 demonstrates the above.

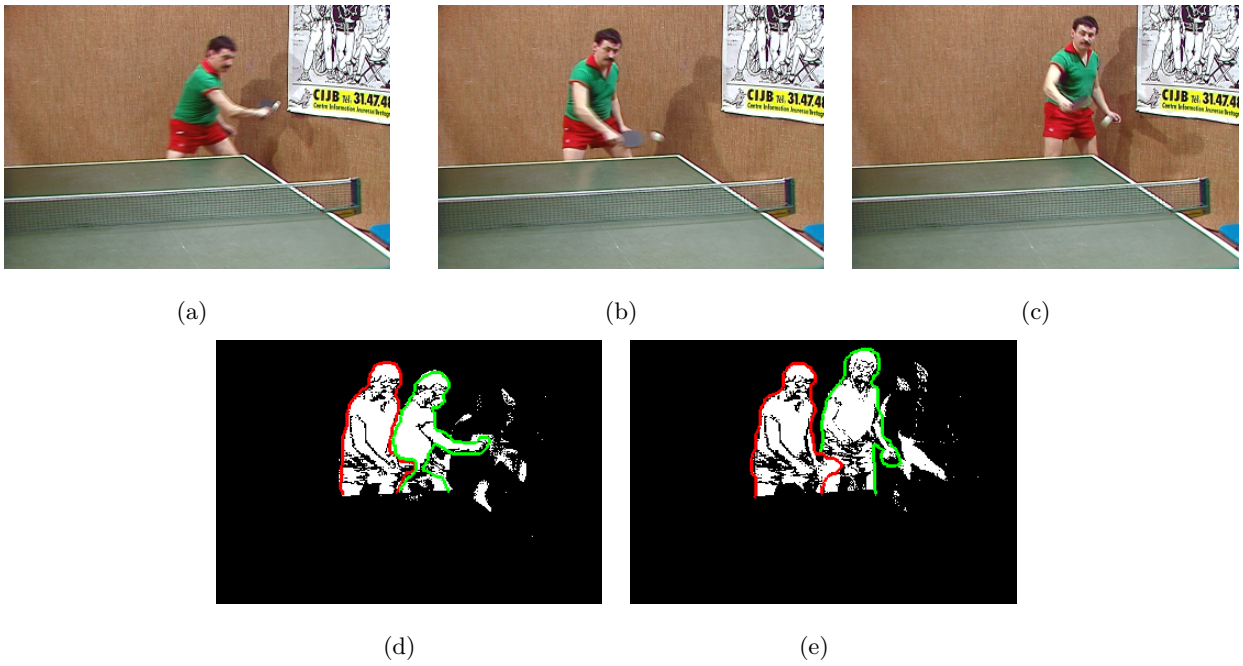


Figure 4.2: (a), (b) and (c) are frame numbers 43,48,52, respectively, taken from the Tennis sequence. (d) and (e) represent the $CDM_{a,b}$ and $CDM_{a,c}$, respectively. The red curves surround the pixels in O_t ($t = 43$). The green curves surround the pixels of O_{t+i} whose locations correspond to the object's locations in (b) and (c).

However, as shown in Fig. 4.2, the CDMs in images 'd' and 'e' have some areas within the MO surface (marked by the red curve) that are considered as 'unchanged' pixels although they were moved. As mentioned, some of the object regions are not sufficiently textured while the change detection algorithm requires a textured area for reliable detection. Therefore, to decrease the dependency of the change detection algorithm on textured areas, we utilize a spatial segmentation in order to find a group of pixels around the regional minima ([20]). These are the boundaries of adjacent regions.

According to the spatial segmentation definition, all the segment boundaries have higher gradient magnitudes than the regional minima. Therefore, it is likely that these boundaries will be sufficiently textured to enable reliable detection of the changes. Thus, each change detection comparison will be applied only to the boundaries of the segments, which are the edges of the RAG, $G = (V, E)$. The results will be assigned as weights to the edges as described below.

Assume that I_t is the reference frame and G represents its RAG after the application of the spatial segmentation. Let $e(u, v) \in E$ be the edge that connects the adjacency nodes (u, v) in the RAG G. Each edge $e(u, v)$ is composed of a nonempty group of pixels. Each edge's pixels are denoted by $pxl^{e(u,v)}(x, y)$ where (x, y) is the coordinate of the edge's pixel and $|e(u, v)|$ is the size of the edge in pixels. Based on [21], we assign a binary value that indicates a change between I_t and I_{t+i} of an edge's pixel such that for all $(x, y) \in e$ we have

$$pxl_{t,t+i}^{e(u,v)}(x, y) = \begin{cases} 1 & \frac{1}{N_\eta} \sum_{k,l \in \eta(x,y)} \left(\frac{I_t(k,l)}{I_{t+i}(k,l)} - \hat{\mu}_{t,t+i}(x, y) \right)^2 > \alpha \\ 0 & \text{else} \end{cases} \tag{4.4}$$

where α is a predefined threshold of the change detection algorithm and $\hat{\mu}_{t,t+i}(x, y)$, which is calculated on a moving squared window $\eta(x, y)$ of $N_\eta = 16$ elements, is:

$$\hat{\mu}_{t,t+i}(x, y) = \frac{1}{N_\eta} \sum_{k,l \in \eta(x,y)} \frac{I_t(k, l)}{I_{t+i}(k, l)}. \tag{4.5}$$

After CD is completed for all the edge pixels in G, we associate with each edge a *Local Change Probability* (LCP) value. The LCP, which is the probability of having a change among the edges of two frames, is given by:

$$lcp_{t,t+i}^{e(u,v)} = \frac{1}{|e(u, v)|} \sum_{x,y \in e(u,v)} pxl_{t,t+i}^{e(u,v)}(x, y) \tag{4.6}$$

for all $e(u, v) \in G$ where $0 < lcp_{t,t+i}^{(e(u,v)} \leq 1,\ i = 1, \ldots, N_t$.

In many applications that find the changes between two images, the threshold determination (Eq. 4.4) is critical for the algorithm's reliability. Generally, this threshold depends on the nature of the local texture in the frame. Weak textured regions have to be considered with a different threshold than strong textured areas. However, since we are interested only in changes of edges in G, which are dominated by higher gradient pixels over the frame, this threshold can be determined only for strong textured areas.

4.4 RAG Weighting Based on the MC Approach

In section 4.3 we defined the multiple comparisons approach between frames that operate on the edges of G. Thus, each comparison between I_t and I_{t+i} provides LCP values to edges that correspond to a certain iteration i, $i = 1, \ldots, N_t$. In this section, we present an approach that maps the local results

$lcp_{t,t+1}^{e(u,v)}, lcp_{t,t+2}^{e(u,v)}, \ldots, lcp_{t,t+N_t}^{e(u,v)}$ to a single value which we call *Global Change Probability* (GCP) that is denoted by $gcp_i^{e(u,v)}$. Each iteration assigns an updated $gcp_i^{e(u,v)}$ to each $e(u,v)$ in G by replacing $gcp_{i-1}^{e(u,v)}$. Thus, $gcp_i^{e(u,v)}$ represents the global change detection probability of $e(u,v)$ for i comparisons. Recall that each $lcp_{t,t+i}^{e(u,v)}$ is the change probability of $e(u,v)$ in I_t relative to I_{t+i} without taking into consideration the changes of I_t and I_{t+i-x} for all x, $t < x < i$. Therefore, the GCP increases the reliability of whether an object's edge has to be considered as a 'change'.

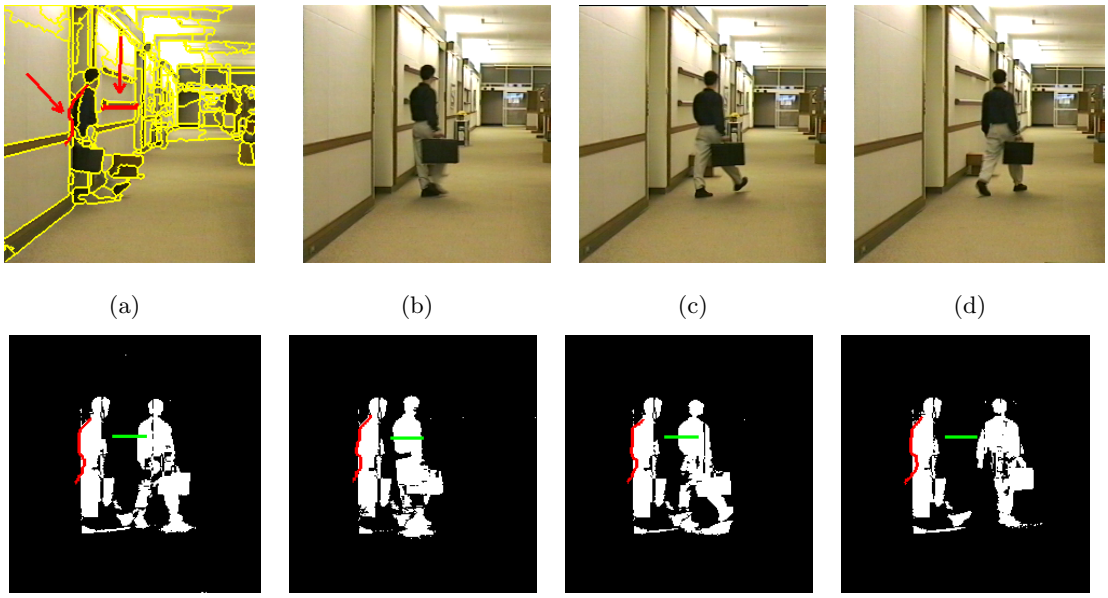
In order to understand how the GCP is computed, we classify the edges in G into three different groups. The first group is called 'object edges', which belongs to the object's region in I_t. The second is called 'occluded edges', which belongs to the background region in I_t, but at least once in the sequence $I_{t+1}, I_{t+2}, \ldots, I_{t+N_t}$, the edge was covered by the object's moving path. The third is called 'background edges', which contains all the edges that belong to the background region in I_t and none of the objects in the sequence $I_{t+1}, I_{t+2}, \ldots, I_{t+N_t}$ cover these edges. The goal of the GCP is to increase the change probability of the 'object edges' at the expense of the other two groups. Its calculation is based on the following assumptions:

'object edges': We assume, based on the MC methodology, that high LCP values of the object edges in I_t will remain relatively close in each additional iteration (comparison) $j, j > i$. In other words, unless the object is relocated to its exact original position in I_t, the LCPs of its regions will remain high.

'occludededges': These edges must be covered at least once during the object's moving path. Therefore, its LCPs have to yield high local values at the i^{th} comparison where the edge is covered by the object's path. Otherwise, it yields a low LCP value. The LCP values of this group usually reach a local maximum when these edges are occluded.

'background edges': These edges are never covered by the object's movement path. Therefore, we assume that low LCP values are expected and the distribution of a set $lcp_{t,t+1}^{e(u,v)}, lcp_{t,t+2}^{e(u,v)}, \ldots, lcp_{t,t+N_t}^{e(u,v)}$ belonging to this group has to be uniform.

The above assumptions are visually demonstrated in Fig. 4.3 followed by its LCP graph calculations presented in Fig. 4.4. Figure 3(a) is the reference frame I_t. The yellow curves are the edges that were taken from the associated RAG after the application of the initial spatial segmentation. Frames 'e','f','g' and 'h' are the CDM results of the above sequence where the red and green curves in each of the CDMs, represent the original edge locations in I_t as marked by the red arrows in Fig. 3(a). As we see in the CDM results (e), (f), (g) and (h), the red curves, which are the 'object edges', are mostly located in the 'change' area. The green edge, which is the 'occluded edge', is both located in the 'change' and 'unchanged' areas according to the object's moving path.



(a) (b) (c) (d)

(e) (f) (g) (h)

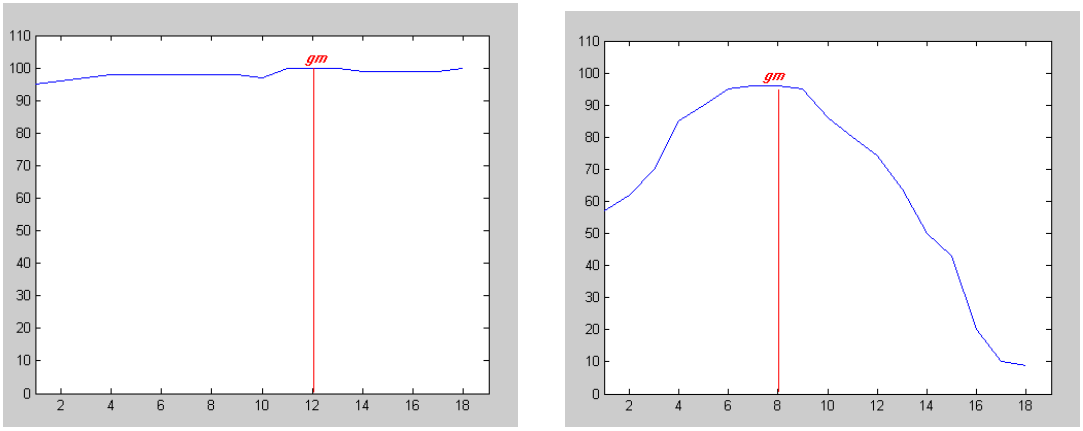
Figure 4.3: (a),(b),(c) and (d) are frames 22,26,30,34, respectively, taken from the video "hall-monitor". (e),(f),(g) and (h) represent the $CDM_{22,26}, CDM_{22,30}, CDM_{22,34}$ and $CDM_{22,38}$, respectively. For example, $CDM_{22,26}$ is the comparison result (for all the image pixels) between frame 22 and frame 26. The output of the initial spatial segmentation is marked by the yellow curves on the reference frame (a).

Frame 22 ($t = 22$) is designated as the reference frame in Fig. 4.3. Figure 4.4 is the graph of $lcp_{22,22+i}^{e(u,v)}$ for the sequence $i = 1, 2, 3, \ldots, 18$, $t = 22$ in Fig. 4.3. Figure 4.4a shows the $lcp_{22,22+i}^{e(u,v)}$ results for $i = 1, 2, \ldots, 18$ of the red edge ('object edge') marked on the CDMs in Figs. 4.3(e, f, g, h). Figure 4.4b shows the $lcp_{22,22+i}^{e(u,v)}$ results for $i = 1, 2, \ldots, 18$ of the green edge ('occluded edge'). As expected from the 'occluded edges' definition, their LCPs reached a global maximum (GM)

$$GM^i \triangleq \max\{lcp_{t,t+i}^{e(u,v)}\}, \quad i = 1, 2, \ldots, N_t \tag{4.7}$$

after $k = 8$ iterative comparisons such that $lcp_{t,t+8}^{e(u,v)} > lcp_{t,t+i}^{e(u,v}$ where $i = k+1, k+2, \ldots, 18$.

In contrast, the $lcp_{22,22+i}^{e(u,v)}$, $i = 1, \ldots, 18$, of the 'object edge' (see Fig. 4.4a) also reached the GM after $k = 12$ comparisons but satisfies $lcp_{t,t+12}^{e(u,v)} \approx lcp_{t,t+i}^{e(u,v)}$, $i = k+1, k+2, \ldots, 18$. It is clear that such typical behavior of the edges, as seen in Fig. 4.4, is independent of the motion type. The differences between sequence types affect only the number N_t of needed comparisons.



(a) (b)

Figure 4.4: The x-axis is the number of i frames that participates in the iterative comparisons. The y-axis is the $lcp_{t,t+i}^{e(u,v)}$ of the i^{th} comparison. (a) is the $lcp_{t,t+i}^{e(u,v)}$ values for $i = 1, 2, \ldots, 18$ of the 'object edge' taken from the CDMs of Fig. 4.3 (the red edge). (b) is the values of the 'occluded edge' taken from the CDMs of Fig. 4.3 (the green edge).

Practically, a reliable classification between the defined groups of edges is difficult to achieve by using pre-defined parameters, which associate each group with a set of independent values. However, the goal of the *MO* extraction algorithm is to distinguish between the 'object edges' and the rest, which we considered as background, rather than finding an appropriate category of parameters for each group.

For that purpose, we map $lcp_{t,t+1}^{e(u,v)}, lcp_{t,t+2}^{e(u,v)}, \ldots, lcp_{t,t+N_t}^{e(u,v)}$ into a $gcp_i^{e(u,v)}$ that is given by:

$$gcp_i^{e(u,v)} = \frac{1}{i}\sum_{j=1}^{i} lcp_{i,t+j}^{e(u,v)} - \sqrt{\frac{1}{i-k+1}\sum_{j=k+1}^{i}\left(GM^i - lcp_{t,t+j}^{e(u,v)}\right)^2} \quad i = 1, 2, \ldots, N_t \tag{4.8}$$

where N_t is the last iterative comparison number and k, $0 < k \leq i$, is the frame index where GM^i (Eq. 4.7). This mapping shifts the outcome value of the 'occluded edges' towards the values of the 'background edges' while preserving the outcome of the 'object edges'. Distinguishing between background edges and occluded edges is irrelevant to the segmentation process since both have to be classified as the same segment.

The computation of Eq. (4.8) relies on the fact that the original location of the 'object edge' stays static after each comparison while the locations of the 'occluded edges' are dynamic and correspond to the object's movements. The mapping in Eq. (4.8) subtracts the LCP mean from the squared error between the GM and each of the successive $lcp_{t,t+i}^{e(u,v)}$, $i = 1, \ldots, N_t$, values. Therefore, when $lcp_{t,t+i}^{e(u,v)}$, $i = 1, \ldots, N_t$, of the object edge are mapped into $gcp_i^{e(u,v)}$, the results will be close to its LCPs mean. This is true since the distribution of $lcp_{t,t+1}^{e(u,v)}, lcp_{t,t+2}^{e(u,v)}, \ldots, lcp_{t,t+N_t}^{e(u,v)}$, which belong to the 'objects edges', is expected to be uniform (see 'object edges' assumption). The same is true when the LCP

set of a background edge $lcp_{t,t+1}^{e(u,v)}, lcp_{t,t+2}^{e(u,v)}, \ldots, lcp_{t,t+N_t}^{e(u,v)}$ is mapped into $gcp_i^{e(u,v)}$ since the 'background edges' LCP distribution is also expected to be uniform (see 'background edges' assumption). Recall that the difference between these two groups of edges is characterized by having a high LCP mean of the object edge and a low LCP mean of the background edge. However, when mapping the LCPs of the 'occluded edges' into $gcp_i^{(e(u,v)}$, the squared errors will be significantly high. This is true since these edges are covered by the object's movement path at least once in the sequence (see 'occluded edges' assumption). Therefore, the outcome will be smaller than its LCP mean. As a consequence, the $gcp_i^{e(u,v)}$ of the 'occluded edges' and the $gcp_i^{e(u,v)}$ of the 'background edges' will be similar (low). Note that the GCP mapping is directly influenced by the number of i comparisons. An automatic procedure to determine the number of needed comparisons will be discussed in section 6.

We assign a $gcp_i^{e(u,v)}$ to each edge $e \in E$ in the RAG $G = (V, E)$. Figure 4.5 shows the edges of the RAG after spatial segmentation was applied to frame I_t taken from the "Tennis" sequence in Fig. 4.2. The gray level intensities represent the GCP values of the edges that were calculated by Eq. (4.8) where $i = 5$. As shown, each 'object edges' in the RAG has higher intensities than the rest of the edges in the frame. The 'occluded edges', which are located in the surrounded area of the object, have higher intensities than the 'background edges' but less than the 'object edges'. Note that from the GCP methodology, the distinction between 'object edges' and 'occluded edges' becomes clearer as long as the object is moving in the sequence.

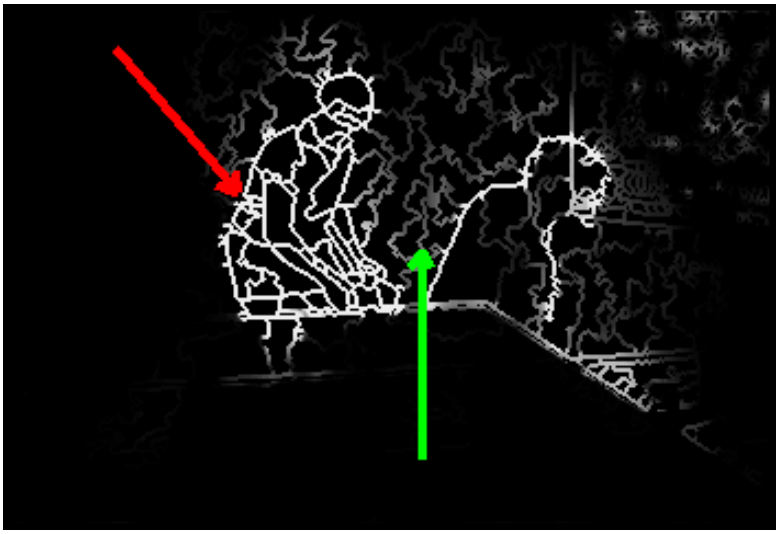


Figure 4.5: The red arrow points to an object area in the reference frame I_t. The green arrow points to an occluded area produced by comparing successive frames with I_t. The second object in I_t is the shadow of the player located on the right side of the frame.

A connectivity analysis between the temporal and the spatial information, which will be discussed in section 5, requires a temporal representation for each node. We now treat a dual problem. Instead

of using edges in the RAG we will use its nodes that represent the segments in the reference frame. To do this, we associate with each node, denoted by v, a gcp_i^v weight which indicates its GCP after i iterative comparisons. The gcp_i^v is based on weighted the mean of the edges that surround each node v. Therefore, the GCP computation of a node v is given by:

$$gcp_i^v \triangleq \frac{1}{|\partial S_v|} \sum_{u_j} \left(gcp_i^{e(u_j,v)} \cdot |e(u_j,v)| \right) \tag{4.9}$$

where v represents a single node in the RAG, $u_j \in V$ is the set of neighboring nodes connected by $e(u_j, v)$, ∂S_v is the set of boundary pixels of v and $|\partial S_v|$ is its size.

5 Connectivity Analysis of Spatio-Temporal information

Previously, we introduced two processes. The construction of the RAG by initial segmentation, and the weight assignment to the RAG edges from the temporal information. In this section, we combine both to extract the *MO*. We describe here an algorithm that extracts the sets of connected nodes that compose the *MO* in I_t. The algorithm is based on the assumption that an *MO* is represented by a set of connected nodes, which produces the highest GCP weight relative to its neighbors. As was mentioned, as i becomes bigger, the GCP's gap between the object's edges and the background's edges is more evident. In general, the algorithm iteratively analyzes the connectivity among the nodes in the RAG while each iteration updates the node weights and the structure of the RAG. After sufficient iterations, the object's set of connected nodes becomes salient and remains unchanged for additional iterations. At this point, the set is extracted. Greedy algorithm searches this group of nodes. The search starts from a single source and spreads to its neighboring nodes. The following describes the node connectivity analysis (NCA) algorithm.

5.1 Notation and Definitions

Let $V_{obj} \triangleq \{v_j \in obj | j = 1, \ldots, |V|, v_j \in V\}$ represents the object regions in I_t and let $V_{back} \triangleq \{v_j \in back | j = 1, \ldots, |V|, v_j \in V\}$ represents the background regions in I_t. Thus, it is clear that the nodes in $G = (V, E)$ of I_t satisfy and $V = V_{obj} \cup V_{back}$ and $V_{obj} \cap V_{back} = \emptyset$.

Each v in G is associated with a gcp_i^v weight where $0 \leq gcp_i^v \leq 100$. Therefore, we can handle the graph $G = (V, E)$ as a topologic surface, which contained 101 levels. Each level is denoted by p. In each level, we consider only the nodes $v \in V$ that satisfy $gcp_i^v \geq p$ and all the edges $e \in v$ that connect these nodes. For example, at level $p = 0$ we consider all the nodes and edges in G. At level $p = 90$ we consider only the nodes gcp_i^v whose weights are above 89. In general, this is similar to a binary image definition, which for a certain threshold in its gray-level image, say $t = 5$, all the pixels that are included will have higher values than $t = 5$.

Definition 5.1. *Given a graph $G = (V, E)$. A set of connected nodes in G at level p, denoted by $SV_k^{(p)},\ k = 1, \ldots |V|$, exists if there is a connected path of edges between each pair of nodes where each node satisfies $gcp_i^v \geq p$.*

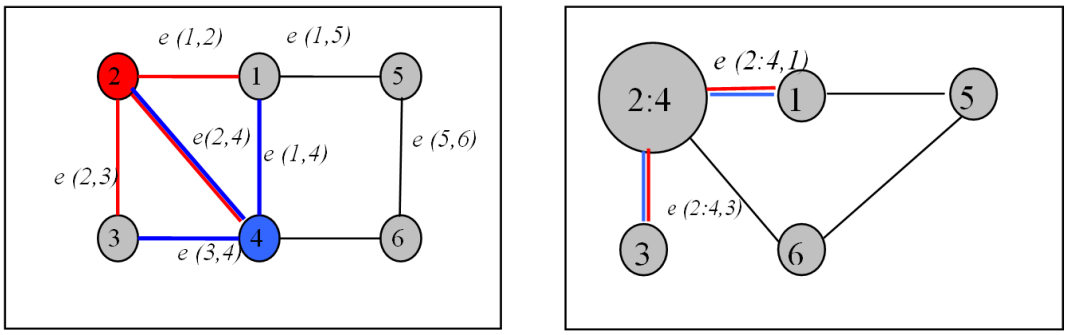
Assume $SV_k^{(p_1)}$ and $SV_l^{(p_2)},\ k, l \in 1, \ldots, |v|$, are two sets of connected nodes in graph G such that $k \neq l$ and $p_1 = p_2$. We required that $SV_k^{(p_1)} \cap SV_l^{(p_2)} = \emptyset$. In addition, for two different levels p_1 and p_2 such that $p_1 < p_2$, either $SV_k^{(p_1)} \supseteq SV_l^{(p_2)}$ or $SV_k^{(p_1)} \cap SV_l^{(p_2)} = \emptyset$ is satisfied. If $SV_k^{(p_1)} \supset SV_l^{(p_2)}$ then we say that the set $SV_l^{(p_2)}$ is a descendant of the set $SV_k^{(p_1)}$. The following defines the process of contracting the edge $e(u, v)$ in G which creates the set $SV_k^{(p)}$ at level p. This process is denoted by $G/e(v, u)$.

Definition 5.2. *Given a graph $G = (V, E)$ and an edge $e = e(u, v) \in E$. The edge contraction in G creates a new graph $G' = (V', E')$ where $V' = V - \{u, v\} + \{\overline{uv}\}$ and $E' = E - \{e(u, v)\} + \{(x, \overline{uv})|$ if $(x, u) \in E$ or $(x, v) \in E$ and $x \neq u, v\}$. $\overline{uv}$ is a new node that is added to the graph.*

In addition to definition 5.2, if $(x, u) \in E$ and $(x, v) \in E$ then we consider only one edge of $e(\overline{uv}, x)$, and recalculate its GCP value by Eq. (5.1). Otherwise, if $(x, u) \in E$ or $(x, v) \in E$ then either $gcp_t^{e(x,u)}$ or $gcp_t^{e(x,v)}$ is assigned, respectively.

$$gcp_i^{e(x,\overline{uv})} = \frac{|e(x, u)| \cdot gcp_i^{e(x,u)} + |e(x, v)| \cdot gcp_i^{e(x,v)}}{|e(x, u)| + |e(x, v)|} \quad i = 1, \ldots, N_t. \tag{5.1}$$

The example in Fig. 5.1 illustrates the contraction of nodes $v = 4$ and $v = 2$ in $G = (V, E)$ (represented by Fig. 5.1a), which is the contraction of the edge $e(2, 4)$. According to definition 5.2, a new graph $G' = (V', E')$ is obtained (represented by Fig. 5.1b) where $E' = E - \{e(2, 4)\} + \{e(1, 2) + e(1, 4) + e(3, 2) + e(3, 4)\}$ and $V' = V - \{v_{(2)}, v_{(4)}\} + \{v_{(\overline{2:4})}\}$. Each set of parallel edges $e(\overline{2:4}, 1)$ and $e(\overline{2:4}, 3)$ (see example in Fig. 1(b)) is considered as a single edge and its GCP is recomputed using Eq. (5.1).



(a) (b)

Figure 5.1: (a) RAG representstion. Node '2' (in red) and node '4' (in blue) are the two contracted nodes. (b) The RAG after the contraction process of $e(2,4)$. Only the GCP of the colored edges in (b) have to be revaluated while the edge $e(2,4)$ in (a) is removed and discarded.

5.2 Nodes Connectivity Analysis Algorithm

Assume $G_i = (V, E)$ is a RAG that is weighted for i iterations. The goal is to extract the sets of connected nodes that have a local GCP maximum relative to their neighbors. This is done by node connectivity analysis which is a greedy-based algorithm. In each comparison, the MC procedure (as was described in section 4.3) assigns weights to the RAG nodes. Thus, in each MC iteration new weights are generated. In this section, we focus on the application of the NCA algorithm that extracts the MO nodes that correspond only to the current i^{th} iteration $i = 1, \ldots, N_t$.

The NCA algorithm handles the RAG $G_i = (V, E)$ as a topologic surface such that $G_i^p = (V^p, E)$ represents a RAG that contains only nodes whose weights are above p where $p = 0, 1, \ldots 100$. In general, the algorithm searches for sets of connected nodes along all p levels of G. Each set, which appears at level p, is being checked whether it is a candidate to be an MO of the current p. After the completion of all searches along the p-levels, the remaining sets of nodes will be the MO for the i^{th} MC.

The first iteration of the NCA algorithm starts with $p = 100$ and contracts all the connected nodes that satisfy $gcp_i^v \geq p$. The obtained sets $SV_k^{(p)}$, $k = 1, \ldots, |v|$, are considered as initial object candidates denoted by $obj_k^{(p)}$. This is needed to create ancestor sets for the next iterative contractions. Then, the weights gcp_i^v of the contracted nodes have to be modified according to the new RAG structure. For example, a set of connected nodes $SV_k^{(p)}$, which appears at level p, is considered as a single node v' with a corresponding $gcp_i^{v'}$ weight. The algorithm repeats this process for $p = 99$. Now the contraction process may either add new nodes to the previous sets $SV_k^{(100)}$ (by contracting $SV_k^{(100)}$ with nodes that satisfy $gcp_i^v = 99$) or create new sets of $SV_k^{(99)}$. After the weights of all the nodes, which are updated at the end of each iteration, each set is checked whether it satisfies the following

definition in order to be considered as an object candidate.

Definition 5.3. *Assume that $SV_k^{(\tilde{p})}$, which appeared at level $\tilde{p}$, is the descendant set of $SV_k^{(p)}$ such that $SV_k^{(\tilde{p})} \subset SV_k^{(p)}$ and $\tilde{p} < p$. The set $SV_k^{(p)}$ is an object candidate at level p if the following conditions hold:*

I. $gcp_i^{SV_k^{(\tilde{p})}} \geq gcp_i^{SV_k^{(p)}}$

II. $gcp_i^{SV_k^{(\tilde{p})}} \geq \alpha$

where gcp_i^v is computed in Eq. 4.9, the set $SV_k^{(p)}$ is considered as a single node v and α is a pre-defined constant.

A set $SV_k^{(p)}$, $k = 1, \ldots, |v|$, which satisfies definition 5.3, is called an object candidate and it is denoted as $obj_k^{(p)}$, $k = 1, \ldots, |v|$. Each $obj_k^{(p)}$ will be constituted a reference ancestor for future considerations to satisfy the object definition 5.3. Each new set, which was obtained at level $\tilde{p}$, is compared to its reference ancestor (object candidate) at p ($p < \tilde{p}$) to verify whether it satisfies definition 5.3. The pre-defined constant α prevents from objects with low GCPs to be considered as candidate objects, and thus, consume unnecessary storage. The algorithm is terminated when the RAG is composed of a single component. Then, the remaining object candidates are the MO segments for the current i^{th} temporal comparison.

The following describes in detail the main steps of the NCA algorithm:

Input:

G_i	The constructed weighted RAG after i iterative comparisons
P_{max}	The maximal GCP of the nodes in G_i
P_{min}	The minimal GCP of the nodes in G_i

Output:

$ObjLst_i$	list of MOs extracted from G_i.

Notation:

$d(v)$	The degree of node v.
$\Gamma(v)$	The set of nodes in G that are adjacent to v.
v'_k	The ancestor of v_i

The NCA process:

1. Construct an array Q of P_{max} entries such that each entry $Q[p]$ will contain a list of nodes v_k in $G_i = (V, E)$ where $gcp_i^{v_k} = p$, $k = 1, \ldots, |v|$. Entry, which does not include any node, will be assigned the NULL pointer.

2. Construct an array B of I entries such that each entry $B[v_k]$ contains a list of $d(v_k)$ nodes which are composed of its adjacent nodes $\Gamma(v_k)$, $k = 1, \ldots, |v|$.

3. **For** $P \leftarrow P_{max}$ to P_{min} **do**

(a) **While** $Q[p]$ is non-empty list **do**

i. $V_k \leftarrow$ Extract/delete a node from $Q[p]$

ii. **While** $B[v_k]$ is non-empty list **do**

A. $V_l \leftarrow$ Extract/delete node from $B[v_k]$

I. CONTRACT[1]$(e(v_k, e_l))$

II. $B[v_k] \leftarrow$UNITE[2]$(B[v_k], B[v_l])$

B. Delete v_l from $Q[p]$

(b) **If** v_k is a descendant of v'_k

i. **If** v_k satisfies the object definition 5.3

A. $(v_k) \leftarrow obj_k^{(p)}$

B. $(v'_k) \leftarrow SV_k^{(p)}$

ii. **Else** $(v_k) \leftarrow SV_k^{(p)}$

(c) **else** classifies v_k as an initial object candidates $obj_k^{(p)}$

(d) Update the GCP values of the contracted nodes (the new sets)

(e) $ObjLst_i \leftarrow obj_k^{(p)}$ while ignoring the initial candidate sets

4. Return $ObjLst_i$

5.3 Step-by-Step Illustration of the NCA Implementation

Figure 5.2 illustrates step-by-step the operation of the NCA algorithm. The given RAG is obtained by the application of a spatial segmentation and weighted by ten MC iterations. The initial RAG (Fig. 2(a)) contains eleven weighted nodes. The number inside each node v indicates its weight gcp_i^v. The blue nodes represent the sets $SV_1^{(75)}$, $SV_3^{(57)}$, $SV_3^{(50)}$, $SV_3^{(46)}$ in Figs. 2(b), 2(d), 2(e) and 2(f), respectively. The red nodes represent $SV_1^{(90)}$, $SV_3^{(75)}$, $SV_3^{(70)}$, $SV_3^{(57)}$ that are considered as $obj_1^{(90)}$, $obj_2^{(75)}$, $obj_3^{(70)}$ and $obj_3^{(57)}$ in Figs. 2(a), 2(b), 2(c) and 2(d), respectively. The red nodes with the bold border (Figs. 2(a), 2(b) and 2(d)) are the initial candidate objects which first appeared in levels $p = 83$, $p = 75$ and $p = 57$, respectively. A red edge indicates that it will be contracted in the next iteration. This example consists of six steps.

Figure 2(a) presents the initial level of the algorithm. Its red node, which appeared at level $p = 83$, is considered as an object candidate due to its first appearance. Then, in Fig. 2(b), the blue node $SV_1^{(75)}$ represents the contraction of the red edge (from Fig. 2(a)). Its updated weight $gcp_i^{SV_1^{(75)}} = 77$ (Fig. 2(b)) is less than its ancestor $gcp_i^{SV_1^{(90)}} = 83$ (shown in Fig. 2(a)), which means that this set

[1]CONTRACT are procedures that operate according to definition 5.2

[2]UNITE are procedures that operate according to definition 5.2

does not satisfy the object definition 5.3. Therefore, it is considered as $SV_1^{(75)}$ and not as $obj_1^{(75)}$. However, its descendant set in the next level (shown in Fig. 2(c)) satisfies the object definition 5.3 since $gcp_i^{SV_3^{(70)}} > gcp_i^{SV_1^{(75)}}$. Thus, this set is considered now as an object candidate $obj_3^{(70)}$ that satisfies definition 5.3 till the completion of all the NCA iterations (Fig. 2(f) shows the last iteration where the graph is composed from a single set $SV_3^{(46)}$). Therefore, it is considered as a single MO in this graph.

In this example, there are three initial object candidates $SV_1^{(90)}$, $SV_3^{(75)}$ and $SV_3^{(57)}$ that appeared in Figs. 2(a), 2(b) and 2(d), respectively. Any descendant from the initial candidates $SV_3^{(57)}$ in Fig. 2(d) did not satisfy the object definition 5.3 in all the successive iterations. Therefore, it is not considered as an MO. In contrast, the two other initial candidates $SV_1^{(90)}$ and $SV_3^{(75)}$ in Figs. 2(a) and 2(b) satisfy the object definition by merging with other sets of nodes, and thus, in this example, they are part of the MO.

In addition, Fig. 5.3 shows the $gcp_i^{SV_k^{(p)}}$ values of three initial candidates $SV_1^{(90)}$, $SV_3^{(75)}$ and $SV_3^{(57)}$ as a function of the number of iterations. The red, blue and green lines represent the $gcp_i^{SV_k^{(p)}}$ values during the NCA iterations of $SV_1^{(90)}$, $SV_3^{(75)}$ and $SV_3^{(57)}$, respectively. The union of the red and blue lines represents the union of its corresponding nodes as shown in Fig. 2(c). This union reached a GM for $gcp_i^{SV_k^{(p)}}$, $p = 0, 1, \ldots, 100$. The green line had no maximal weight satisfying object definition 5.3. Thus, it is not considered as $obj_k^{(p)}$ at any level p.

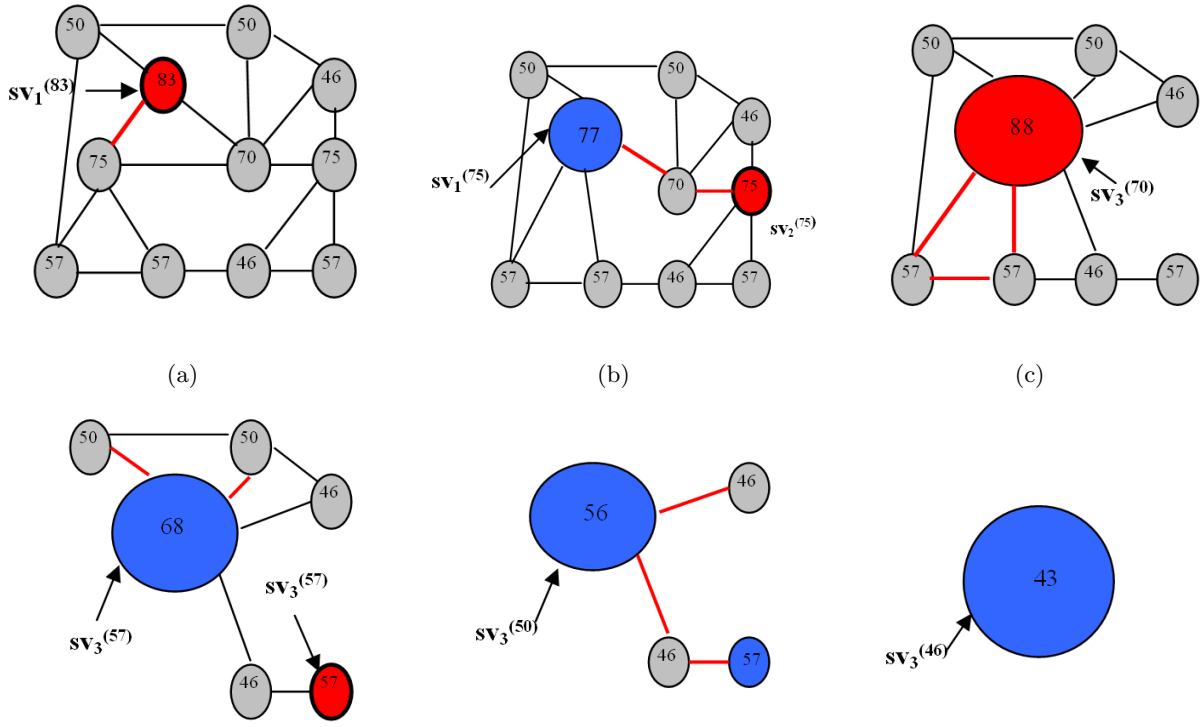


Figure 5.2: Each graph represents a single iteration of the NCA algorithm. The red nodes, which are bounded by a bold border, are the initial object candidates. The blue nodes are the contracted set of connected nodes. The red node in (c), which appeared at $p-$level=70, reached a maximum weight ($gcp_{10}^{SV_3^{(70)}} = 88$) among all the p levels. Thus, it satisfies object definition 5.3

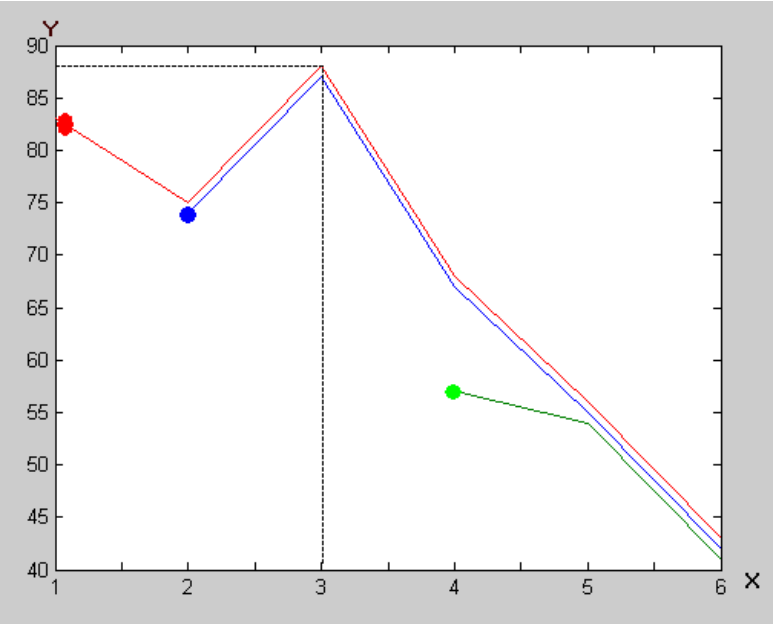


Figure 5.3: GCP values of three initial object candidates. The red, blue and green lines represent the initial candidate nodes from Figs. 2(a), 2(b) and 2(d), respectively. The x-axis represents the number of NCA iterations and the y-axis represents the node weights. The dashed black line points to the GM of the two united sets of nodes.

6 On the NCA Application during Temporal Iterations

A description of the NCA algorithm for extracting the MO sets per a single MC iteration was given in section 5. Each of the extracted sets by the NCA application corresponds to a $G_i = (V, E)$ that is weighed after $i = 1, \ldots, N_t$, iterations. Therefore, we have to determine the minimal number of N_t iterations in which the extracted sets accurately represent the objects in I_t. The proposed algorithm is gradually evolving. Therefore, as i becomes bigger, the extracted set of the NCA reflects the MO more accurately. A naive approach calls for the application of the NCA till a scene-cut is reached, which means a maximum number of iterations are used. This is not practical for real-time considerations. An automatic adaptive determination of N_t is presented while preserving low computational load.

As mentioned in section 4.4 and was illustrated in Fig. 4.4, the 'object edges' are characterized by having almost constant weights after reaching the global maximum. Therefore, we anticipate that in a certain iteration in the temporal phase, say x, the moving object set of nodes will remain similar in shape to every NCA application where $i > x$. Thus, we suggest to compare each set extracted by the NCA in the i^{th} temporal iteration, to its overlapped set, extracted in the previous $(i-1)^{th}$ iteration. Object sets, whose shapes remain similar for γ consecutive NCA applications, are considered as reliable MO sets and no more temporal comparisons are required. Note that as γ decreases the

probability to miss detect the exact boundaries of the MO increases and vice versa.

In order to determine shape similarities between obj_i and obj_{i-1} extracted from the i^{th} and $(i-1)^{th}$ NCA applications, respectively, we define next a distance measure, denoted as $D(\partial obj_i, \partial obj_{i-1})$, between the digital curves of two object boundaries ∂obj_i and ∂obj_{i-1}.

Definition 6.1. *A distance of a given pixel $a \in \partial obj_i$ from a curve ∂obj_{i-1}, denoted by $d(a_i, \partial obj_{i-1})$, is the distance between a_i and the nearest point to a_{i-1} in ∂obj_{i-1}.*

Definition 6.2. *A distance of a given curve ∂obj_i from a curve ∂obj_{i-1}, denoted by $d(\partial obj_i, \partial obj_{i-1})$, is the sum of square distances between the pixels of ∂obj_i and ∂obj_{i-1}.*

Thus, the distance between two curves is calculated by:

$$d(\partial obj_i, \partial obj_{i-1}) \triangleq \frac{1}{|\partial obj_i|} \sum_{k=1}^{|\partial obj_i|} (d(a_k, \partial obj_{i-1})^2 \tag{6.1}$$

where $|\partial obj_i|$ is the number of pixels in the curve ∂obj_i. However, the distance $d(\partial obj_i, \partial obj_{i-1})$ is not necessarily equal to the distance $d(\partial obj_{i-1}, \partial obj_i)$. Therefore, the distance $D_i(\partial obj_i, \partial obj_{i-1})$ is calculated by:

$$D_i(\partial obj_i, \partial obj_{i-1}) = \max\{d(\partial obj_i, \partial obj_{i-1}), d(\partial obj_{i-1}, \partial obj_i)\}, \quad i = 1, \ldots, N_t. \tag{6.2}$$

We say that obj_i is "stable" (reaching steady state) if the distance between the extracted objects for $\gamma = 1, \ldots, i-1$, NCA iterations decrease. Thus, a given obj_i is a stabilized object if it satisfies the following for γ NCA iterations:

$$stbl_{obj_i}^{\gamma,i} \triangleq |D_j(\partial obj_j, \partial obj_{j-1} - D_{j-1}(\partial obj_{j-1}, \partial obj_{j-2}| \leq \epsilon \quad j = \gamma + 1, \ldots, i. \tag{6.3}$$

The "stability" calculation (Eq. 6.3) is examined per NCA iteration for each extracted object. If $stbl_{obj_i}^{\gamma,i} < \epsilon$, $\gamma = 1, \ldots, i-1$, $i = 1, \ldots, N_t$, we classify this object as stable and no additional NCA iterations are needed. We considered a stable object as an accurate representation of the MO in I_t.

The NCA application is terminated when all the object sets are stabilized. Hence, we are able to determine how many iterative comparisons N_t are needed in order to extract each MO independently of the other objects in the frame. Note that the "stability" computation (Eq. 6.3) is strongly influenced by both the video nature (camera noise and luminance change) and the object's motion type (fast or slow). Our experiments (section 7) will show that in fast object's motion, *stbl* becomes smaller than ϵ after a few NCA iterations while in a slow object's motion, more NCA iterations are needed.

7 Experimental Results

The proposed algorithm was applied to three different video sequences, which are characterized by their different nature. Each frame in the following experiments presents the NCA result after the completion of a single iteration. The last frame in each experiment (marked by a green border) represents the final output of the segmentation algorithm that is obtained after stability is achieved. As we will show, different types of sequences do not affect the accuracy of the algorithm. They affect only the number of required iterations.

Figure 7.1 presents the segmentation of one frame taken from the "Silence" video sequence. Each frame 1(a)-1(h) is the output of a single NCA application. The yellow polygons are the boundaries of the MO set (segment) extracted after each NCA application. The red curves are the boundaries of the initial spatial segmentation. This sequence is characterized by having a slow-moving object with insignificant luminance changes in the background region. The output from the first iteration is given in frame 1(a). After the completion of six iterations (shown in frame 1(f)), the algorithm succeeded in separating between the MO and the background. Frames 1(f)- 1(h) show that the segmented object remains stable and no additional iterations are needed.

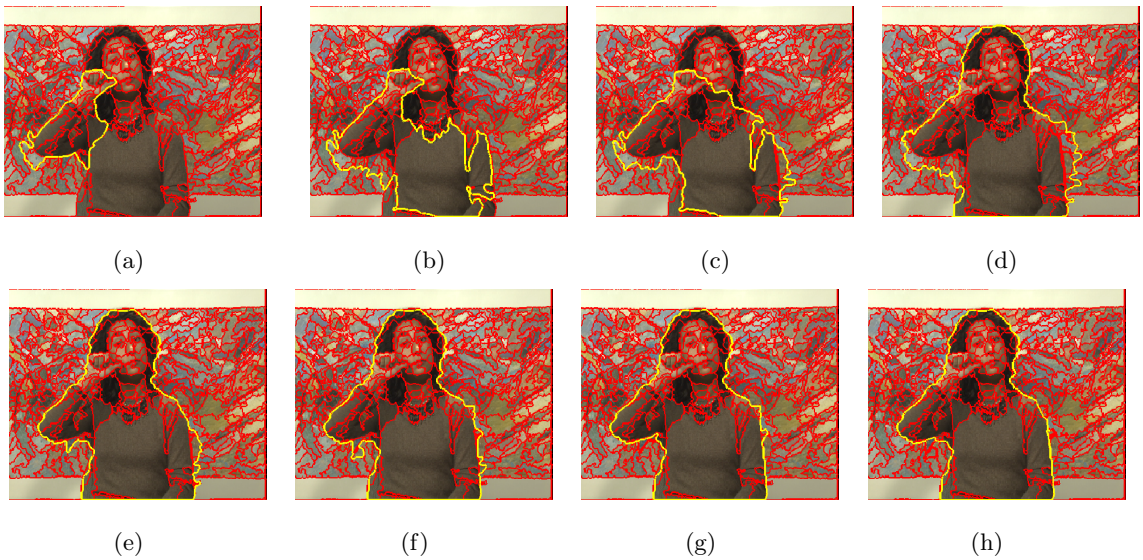
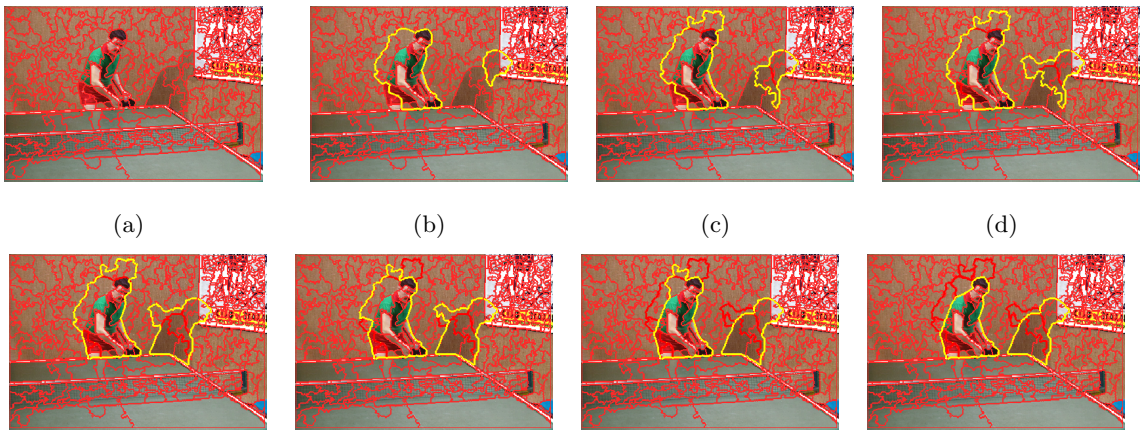


Figure 7.1: Temporal results of the segmentation algorithm where (h) is the final output of frame 48 taken from the "Silence" video sequence. Eight iterations of the NCA were needed to reach the final segmentation.

Figure 7.2 presents the segmentation of one frame taken from the "Tennis" video sequence. Each frame (a)-(h) presents the output of a single NCA application. The yellow polygons are the boundaries of the MO sets (segment) as extracted after each NCA application. The red curves are the boundaries of the initial spatial segmentation. This sequence is characterized by a fast-moving object in contrast to

the previous example. The output from the first iteration is given by frame 2(a). After the completion of six NCA applications (shown in frame 2(h)), the algorithm succeeded in separating between the MO and the background. It is easier to detect fast-moving objects than slow moving ones, but it is more difficult to separate between occluded and object regions. In this example, the object was detected after the first iteration (frame 2(a)) but the separation from the occluded regions was achieved after six iterations (frame 2(f)). Frames 2(f)- 2(h) show that the segmented object remains stable and no additional iterations were needed.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 7.2: Temporal results of the segmentation algorithm where (h) represents the final output of frame 10 taken from the "Tennis" video sequence.

Figure 7.3 presents the segmentation of one frame taken from the "Hall Monitor" video sequence. Each image (a)-(j) presents the output from a single NCA application. The yellow polygons are the boundaries of the MO set (segment) as extracted after each NCA application. The red curves are the boundaries of the initial spatial segmentation. This sequence is characterized by relatively fast-moving objects with significant luminance changes among consecutive frames in contrast to the two previous examples. The output from the first iteration is given by frame 7.3a. Since this sequence has noise and flickering from luminance changes, the algorithm needed more MC iterations to accurately detect the MO than in the previous example. However, this example demonstrates that the nature of the sequence affects only the required number of iterations. The object was accurately detected after eight iterations (shown in frame 7.3h) and remained stable in additional iterations. Stability was achieved after ten iterations (illustrated by frame 7.3j).

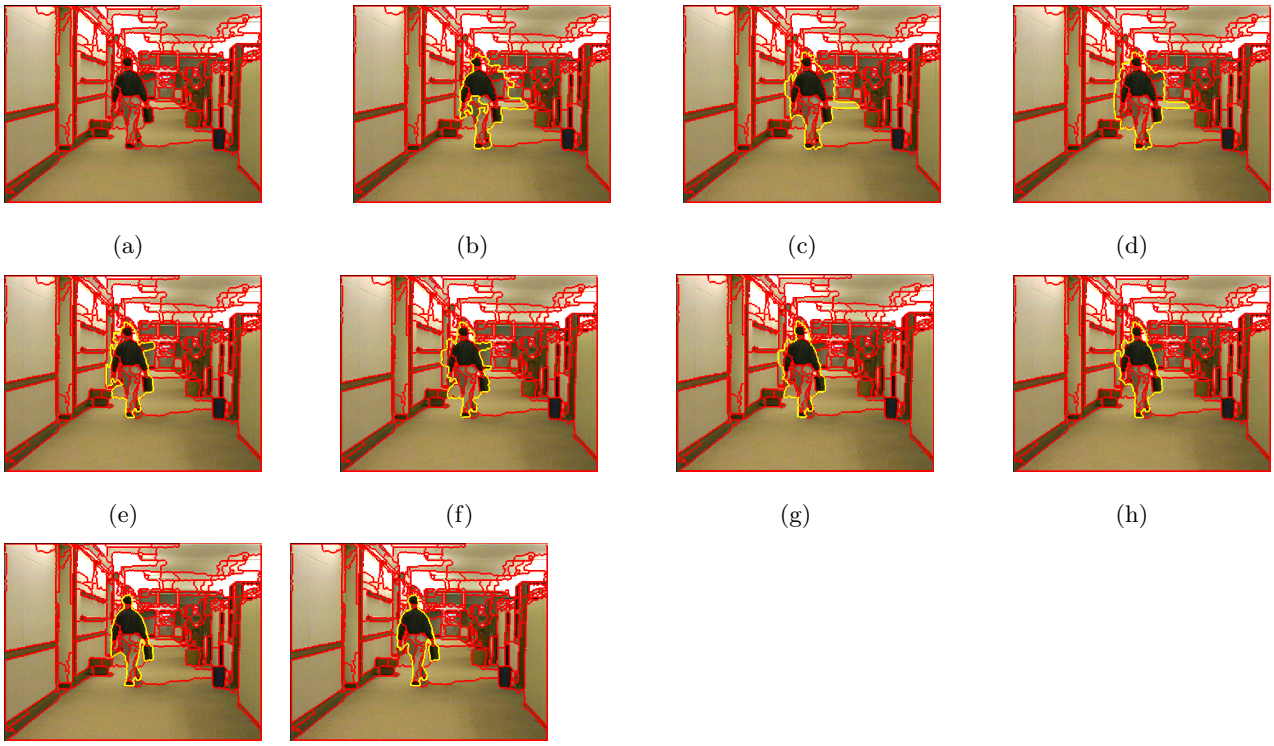


Figure 7.3: Temporal results of the segmentation algorithm where (j) represents the final output of frame (a) (the 62th frame) taken from the "Hall Monitor" video sequence.

8 Conclusions and Discussion

This paper presents a robust algorithm for moving object segmentation. The performance of the algorithm is directly dependent on the allowed number of iterations. When no time limitation is imposed, the algorithm provides a reliable and accurate segmentation.

The algorithm is divided into two steps. The first is a spatial segmentation, which partitions the input frame into semantic homogeneous regions, where each region is distinguished by its encompassing boundaries, obtained from the segmentation process. The output of this step is an RAG where the nodes represent the homogeneous regions and the edges represent the adjacency of the regions. The second step is composed of two phases, which operate in parallel to each other. The first is the iterative temporal comparisons between successive frames with a single reference frame. The second analyzes the nodes' connectivity where the edges represent temporal information. The algorithm is terminated when the extracted object's shape becomes stable.

The algorithm handles well different types of video sequences. No predefined parameters are given, and no prior information is needed. The algorithm's performance is independent of the nature of the video sequences. In the case where the algorithm has a difficulty in determining which area in the

frame is the MO, it proceeds by adding temporal information to the nodes' connectivity analysis phase till the extracted objects in each iteration become stable relative to their previous iterations.

However, when the video sequence contains slow-moving objects with luminance changes and noise, the algorithm will need more comparisons to extract the MO in the reference frame. Then, the delay between the reference frame and the completion of the segmentation process may be too long for real time considerations. One option to reduce this delay as a future work is to incorporate a tracking mechanism ([1]), which preformed on two consecutive frames, into the proposed algorithm. Such integrated will enable to follow the MO without repeatedly calling the algorithm to extract the MOs.

References

[1] O. Miller, A. Averbuch, E. Navon, "Tracking of moving objects in video through invariant features in their graph representation", to appear in EURASIP Journal on Image and Video Processing, special issue on "Video Tracking in Complex Scenes for Surveillance Applications".

[2] A. Averbuch, Y. Keller, "Fast motion estimation using bidirectional gradient methdos", IEEE Trans. on Image Processing, 13(8), pp. 1042-1054, 2004.

[3] A. Averbuch, Y. Keller, "Fast gradient methods based global motion estimation for video compression", IEEE Trans. on Circuits and Systems for Video Technology, 13:4, 300-309, 2003.

[4] MPEG-4 Video Group, "MPEG-4 video verification model version 7.0,"ISO/IEC JTC1/SC29/WG11, MPEG97/N1642", Bristol, England, Apr. 1997.

[5] M. M. Yeung, B. L. Yeo, W. Wolf, and B. Liu, "Video browsing using clustering and scene transitions on compressed sequences," *Proc. SPIE*,vol. 2417, pp. 399-413, 1995.

[6] M.Hotter, R.Thoma, "Image segmentation based on object oriented mapping parameter estimation", *Signal Processing*, Vol. 15, No. 3, pp.315-334, October 1988.

[7] J.Y.A. Wang and E.H. Adelson, "Spatio-temporal segmentation of video data", *Proc. Of the SPIE: image and video Processing II*, Vol.2182, February 1994.

[8] J. Shi, J. Malik, "Motion Segmentation and Tracking Using Normalized Cuts", *University of California*, Berkeley Report No. UCB/CSD-97-962 June 1997.

[9] F. Dufaux, F. Moscheni and A. Lippman, "Spatio-temporal segmentation based on motion and static segmentation", *IEEE Proc. Int. Conf. Image Processing 95*, Washington, DC October 1995.

[10] M.M Chang, M.I. Sezan and A.M. Tekalp, "An algorithm for simultaneous motion estimation and segmentation", *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Adelaide, Australia, Vol. V, PP. 221-224, April 1994.

[11] M. Kim, J. G. Choi, D. Kim, H. Lee, M. Lee, C. Ahn, and Y.-S. Ho, "Automatic Segmentation of Moving Objects in Image Sequences Based on Spatio-Temporal Information", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No 8, pp. 1216–1226, December 1999.

[12] T. Meier, K. N. Ngan, "Video Segmentation for Content-Based Coding", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No. 8, pp. 1190–1203, December 1999.

[13] C. Kim, J-N. Hwang, "Fast and Automatic Video Object Segmentation and Tracking for Content-Based Applications", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, No. 2, pp. 122–129, February 2002.

[14] D. Wang, "Unsupervised Video Segmentation Based on Watersheds and Temporal Tracking", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 8, No. 5,pp. 539–546, September 1998

[15] [12] Hai Gao, Wan-Chi Siu, and Chao-Huan Hou, "Improved Techniques for Automatic Image Segmentation", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, No. 12, pp. 1273–1280, December 2001.

[16] Y. Yokoyama, Y. Miyamoto, and M. Ohta, "Very low bit rate video coding using arbitrarily shaped region-based motion compensation," *IEEE Trans. Circuits Syst. Video Technology*, vol. 5, pp. 500–507, Dec.1995.

[17] L. Wu, J. Benoise-Pineau, Ph. Delagnes, and D. Barba, "Spatio-temporal segmentation of image sequences for object-oriented low bit-rate image coding," *Signal Processing: Image Communication*, vol. 8, pp. 513–543, 1996.

[18] J. G. Choi, M. Kim, M. H. Lee, and C. Ahn, "Automatic segmentation based on spatio-temporal information,", ISO/IEC JTC1/SC29/WG11 MPEG97/m2091, Bristol, U.K., Apr. 1997.

[19] K. Haris, S. N. Efstratiadis, N. Maglaveras and A. K. Katsaggelos, "Hybrid image segmentation using watershed and fast region merging,", *IEEE Trans. Image Processing*, vol. 7, pp. 1684–1699, Dec. 1998.

[20] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 583–598, 1991.

[21] K. Skifstad and R. Jain, "Illumination Independent Change Detection for Real World Image Sequences", *Computer Vision, Graphics, and Image Processing*, Vol. 46, pp. 387–399, 1989.

[22] D. Gatica-Perez, C. Gu, and M. Sun, "Semantic Video Object Extraction Using Four-Band Watershed and Partition Lattice Operators", *IEEE Trans. Circuits Syst. Video Technology*, vol. 11, No. 5, pp. 603–618, May 2001.

[23] F. Dufaux and J. Moscheni and A. Lippman, "Efficient, robust and fast global motion estimation for video coding", *IEEE Trans on image processing*, Vol .9, Mo. 3, pp. 497–500, 2000.

[24] T. Corman, C. Leiserson, R. Rivest, Introduction to Algorithms, MIT press, 1990.